Trusted Clouds

Mike Burmester

Florida State University

DIMACS/BIC/A4Cloud/CSA International Workshop on Trustworthiness, Accountability and Forensics in the Cloud (TAFC)

June 6, 2013

Talkthrough

1. Motivation
2. Cloud architectures
3. Trusted Computing architectures
4. Threat model and security framework for TC-compliant systems
5. An architecture for trusted clouds
6. The dark side of clouds

Motivation

Scenario :

A critical infrastructure network or DoD type network.
Service endpoints are lightweight devices (e.g., tablets) with practically no nonvolatile memory to minimize exposure to theft.

Requirement:

Protect the communication channels.

- A cloud deployment that supports secure computing services.
- The deployment should enforce:
  - need-to-know and separation-of-duties policies.

Cloud architectures

- The Cloud is a client-driven access control infrastructure that manages computing services.
- A cloud monitor mediates between clients and service providers with access granted based on service agreements that establish a trust-bond between clients and providers.

Cloud monitors

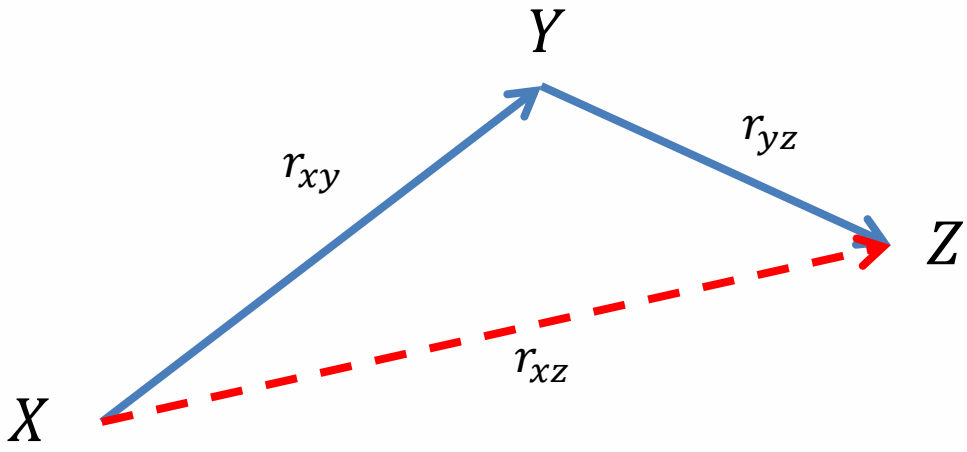
A cloud monitor can be modeled by a directed trust-graph $G = (V, E)$ with nodes the clients, the providers and services.

There are two types of edges:

a) Edges $X \xrightarrow{r_{xy}} Y$ that link clients to providers with labels r_{xy} that:
   - contain a SLA, terms of use, privacy/security policies and the compensation in the event the provider fails to deliver at the specified level or violates agreed policies;
   - capture the confidence the client has in the provider regarding specific services as well as the risks involved.

b) Edges $Y \xrightarrow{r_{yz}} Z$ that link providers to services with labels r_{yz} that:
   - contain the agreement regarding the particular service, the security policies and the compensation if the service is not delivered at the specified level or agreed policies are violated.
   - capture the confidence that the client has in the specific service.

Trust is not necessarily transitive



- The trust-edge XZ is only defined if: $r_{xy}\ dom\ r_{yz}$.
- The trust-graph G is dynamic with edges added (deleted) in real-time corresponding to new service requests or new services becoming available (or withdrawn).

Access Control policies

- Traditional access control models focus on managing data resources, *not services that are functions of data.*
- For cloud deployments we have to design *new models* and specify *new policies* for the secure management of services.
- We propose to use the trust levels r_{xy} as discussed earlier, modified appropriately to capture dynamic management and policies that enforce need-to-know and separation-of-duties.
  - Need-to-know refers to the trust r_{xz} needed for accessing service Z.
  - Separation-of-duties refers to the trust r_{yz} between a provider and service Z.
- These should be no more than strictly necessary.

Trusted Computing (1/3)

- The Trusted Computing Group published specifications for architectures and interfaces for several computing implementations to meet the functional and reliability requirements of computer systems and provide increased assurance of trust.
- Two such architectures are the Trusted Platform Module (TPM) and the Trusted Network Connect (TNC).

Trusted Computing (2/3)

- The TPM is a Trusted Computing (TC) architecture that establishes trust in the expected behavior of a system.
  It has two basic capabilities: *remote attestation* and *sealed storage*.
- Trust is based on an integrity protected boot process in which executable code and associated configuration data are measured before execution.
  - a hash of the BIOS code is stored in a Platform Configuration Register (PCR).
- Sealed storage is used to protect cryptographic keys.
- The TPMs must be physically protected from tampering.
  This includes binding the TPM to physical parts of the platform.

Trusted Computing (3/3)

- The TNC is an architecture for trusted network applications. A trusted link between a client and server is established only if:
  - the identity of the client and server is trusted;
  - the client has real-time access to the server;
  - the client and server are authenticated;
  - the integrity of communicated data, and if necessary the confidentiality, is enforced by the TPM.
- The TC paradigm has been studied extensively, with TPM- and TNC-compliant systems implemented in several configurations.

Threat model for TC-compliant systems

- The TPM prevents compromised components of a TC-compliant system from executing.
- As a result, if we *exclude run-time (execution) threats*, malicious threats are reduced to DoS threats.
- There are two kinds of faults that may affect a TC-compliant computer system: natural and adversarial.
  - Natural faults can be predicted, in the sense that an upper bound on the probability of such faults can be estimated. Redundancy can then be used to reduce this probability to below an acceptable threshold.
  - Malicious DoS faults cannot be predicted. However they are overt and, because of the TPM and TNC integrity verification, must be physical (e.g., involve tampering the TPM chip).

Security framework for TC-compliant systems

So there is a cost involved.

- One way to thwart them is to make the cost high enough to prevent them.
- There are several security models that use economics and risk analysis based on redundancy that are appropriate for threat models with overt faults.
- These assume a bound on adversarial resources and an architecture with sufficient redundancy to make such DoS attacks prohibitively expensive.

The good, the bad and the ugly

The good

The TPM/TNC platforms protect system components from behaving in an unexpected way.

The bad

Only trusted code can execute.

Therefore the integrity of trusted code is a fundamental requirement.

The system software must be well designed, with no security holes backdoors or vulnerabilities that could be exploited by an adversary.

An exploit in the OS may allow the adversary to bypass the protection offered by the TPM.

The good, the bad and the ugly

The ugly

Proof-carrying code is not closed with respect to composability unless the proofs are composable. Consequently the TPM provides integrity guarantees only at *load-time*, not *run-time*.

There are several run-time attacks use metamorphic malware such as the self-camouflaging Frankenstein or more generally, return oriented programming (ROP).

For these the adversary must be able to control the execution on the stack, and there are (rather costly) ways to prevent this.

An architecture for trusted clouds

The basic components of a Cloud are: the clients, the providers, the computing services and the cloud Monitor.

For a Trusted Cloud we propose an architecture with:

- A private cloud deployment and trusted service providers.
- A trusted cloud monitor.
- An access control model for computing services that supports need-to-know and separation-of-duties policies.
- TC-compliant computing services.
- Lightweight TC-compliant client service endpoints.

The dark side of clouds

- By assuming that all service providers are trusted, and that computing services will not deviate from their expected behavior, we make certain that the only remaining threats are those that bypass the trust mechanisms.
- In particular threats resulting from concurrent execution of trusted code. To mitigate such threats, any successful approach will have to:

a) limit the openings for exploitation on platform software, e.g., operate within a well-defined environment of sets of duties;

b) employ methods to detect run-time compromise.

Any questions?